

R para iniciantes

Aula III Leitura e Manipulação de dados

Carlos Henrique Tonhatti

Dúvidas da última aula?

Sumário

- 1 Entrando e saindo com dados do R
 - Importação de dados
 - Exportando dados

Sumário

1 Entrando e saindo com dados do R

- Importação de dados
- Exportando dados

2 Manipulando dados

- Edição
- Criação de subconjuntos
- Criação de filtros
- Ordenação

Sumário

1 Entrando e saindo com dados do R

- Importação de dados
- Exportando dados

2 Manipulando dados

- Edição
- Criação de subconjuntos
- Criação de filtros
- Ordenação

Formas de entrada de dados.

Existem algumas formas de entrar com dados no R:

- Digitando diretamente no terminal (como foi feito nas últimas aulas).
- Importando os dados de outras fontes (arquivos, internet).

Sumário

1 Entrando e saindo com dados do R

- Importação de dados
- Exportando dados

2 Manipulando dados

- Edição
- Criação de subconjuntos
- Criação de filtros
- Ordenação

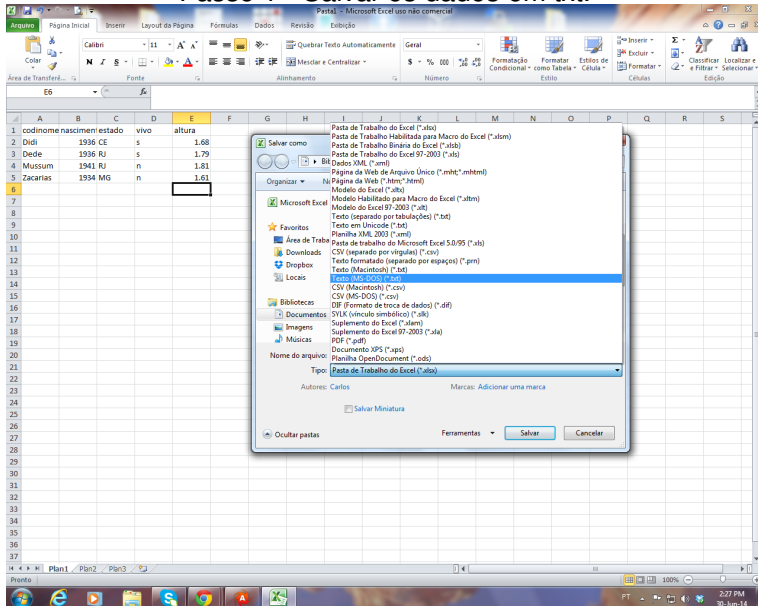
Importação de dados

A importação de dados no R depende do tipo e formato dos dados que se pretende importar. A função genérica `read.table` importa dados em formato de tabela.

Importação de dados

```
read.table(file, header = FALSE, sep = , quote = "\"",  
           dec = ".", row.names, col.names,  
           as.is = !stringsAsFactors,  
           na.strings = "NA", colClasses = NA, nrows = -1,  
           skip = 0, check.names = TRUE, fill = !blank.lines.skip,  
           strip.white = FALSE, blank.lines.skip = TRUE,  
           comment.char = "#",  
           allowEscapes = FALSE, flush = FALSE,  
           stringsAsFactors = default.stringsAsFactors(),  
           fileEncoding = , encoding = "unknown", text)
```


Passo 1 - Salvar os dados em txt.



Exemplo

Passo 2 - Salvar o arquivo na pasta de trabalho do R.

Exemplo

Passo 2 - Salvar o arquivo na pasta de trabalho do R.

Exemplo

Passo 2 - Salvar o arquivo na pasta de trabalho do R.

Passo 3 - Importar o arquivo

```
> trapa<- read.table("trapa.txt",header= TRUE, sep= " ",  
dec= ".")
```

Passo 4- Verificar o que foi importado

```
> trapa
```

	codinome	nascimento	estado	vivo	altura
[1,]	Didi	1936	CE	s	1.68
[2,]	Dede	1936	RJ	s	1.79
[3,]	Mussum	1941	RJ	n	1.81
[4,]	Zacarias	1934	MG	n	1.61

Considerações

- Os passos do exemplo podem variar devido à complexidade dos dados;
- O objeto criado é um `data.frame`;
- Importante é entender como os dados estão organizados antes de importar;
- A função `read.table` é flexível. Pode ser ajustada a vários formatos de dados.

Considerações

- O passos do exemplo podem variar devido á complexidade dos dados;
- O objeto criado é um `data.frame`;
- Importante é entender como os dados estão organizados antes de importar;
- A função `read.table` é flexível. Pode ser ajustada á vários formatos de dados.Principais argumentos:
 - `file` nome do arquivo o qual tem os dados,
 - `header` indica se a primeira linha contem os títulos das colunas,
 - `sep` indica qual o separador de campos,
 - `dec` indica qual simbolo é usado para separar decimais.

Considerações

- O passos do exemplo podem variar devido á complexidade dos dados;
- O objeto criado é um `data.frame`;
- Importante é entender como os dados estão organizados antes de importar;
- A função `read.table` é flexível. Pode ser ajustada á vários formatos de dados.Principais argumentos:
 - `file` nome do arquivo o qual tem os dados,
 - `header` indica se a primeira linha contem os títulos das colunas,
 - `sep` indica qual o separador de campos,
 - `dec` indica qual simbolo é usado para separar decimais.

```
read.table("trapa.txt",header= TRUE, sep= " ", dec= ".")
```

Considerações

- O passos do exemplo podem variar devido á complexidade dos dados;
- O objeto criado é um `data.frame`;
- Importante é entender como os dados estão organizados antes de importar;
- A função `read.table` é flexível. Pode ser ajustada á vários formatos de dados.Principais argumentos:
 - `file` nome do arquivo o qual tem os dados,
 - `header` indica se a primeira linha contem os títulos das colunas,
 - `sep` indica qual o separador de campos,
 - `dec` indica qual simbolo é usado para separar decimais.

```
read.table("trapa.txt",header= TRUE, sep= " ", dec= ".")
```

Mais informações em `?read.table`

Dicas

Estabeleça um padrão para os seus dados:

Dicas

Estabeleça um padrão para os seus dados:

- Qual separador de campo (TAB, “,”, “;”)?
- Qual símbolo para decimais (“.”, “,”)?
- Nome das variáveis na colunas?

Dicas

Estabeleça um padrão para os seus dados:

- Qual separador de campo (TAB, “,”, “;”)?
- Qual símbolo para decimais (“.”, “,”)?
- Nome das variáveis na colunas?

Confira se os dados foram lidos de forma correta. Use os comandos `file`, `str()`, `dim()`, `head()`, `tail()`

Sumário

1 Entrando e saindo com dados do R

- Importação de dados
- Exportando dados

2 Manipulando dados

- Edição
- Criação de subconjuntos
- Criação de filtros
- Ordenação

Exportando dados do R

A exportação de dados do R pode ser feita de várias formas. A função genérica `write.table` exporta os dados em forma de *data frame* criando um novo arquivo.

Exportação de dados

```
write.table(x, file = " ", append = FALSE, quote = TRUE, sep = " ",  
           eol = "\n", na = "NA", dec = ".", row.names = TRUE,  
           col.names = TRUE, qmethod = c("escape", "double"),  
           fileEncoding = " ")
```

```
> write.table(trapa, file="trapa2.txt")
```

Sumário

1 Entrando e saindo com dados do R

- Importação de dados
- Exportando dados

2 Manipulando dados

- Edição
- Criação de subconjuntos
- Criação de filtros
- Ordenação

Sumário

1 Entrando e saindo com dados do R


- Importação de dados
- Exportando dados

2 Manipulando dados

- Edição
- Criação de subconjuntos
- Criação de filtros
- Ordenação

Edição

A função `fix()` abre o objeto dentro de uma planilha possibilitando pequenas edições.



	nomes	ano.nasc	vive
1	Didi	1936	V
2	Dedé	1936	V
3	Mussum	1941	F
4	Zacarias	1934	F

Edição: nomes de colunas e linhas

Nomes das colunas e linhas

Retorna o nome das colunas

`names(x)`

`colnames(x)`

Retorna o nome das linhas

`rownames(x)`

Exemplo

```
> names(trapa)
[1] "codinome" "nascimento" "estado" "vivo" "altura"
```

É possível alterar o nome das colunas ou linhas usando a mesma função:

```
> names(trapa) <- c("COD", "NASC", "UF", "VIVO", "Altura")
```

	COD	NASC	UF	VIVO	Altura
[1,]	Didi	1936	CE	s	1.68
[2,]	Dede	1936	RJ	s	1.79
[3,]	Mussum	1941	RJ	n	1.81
[4,]	Zacarias	1934	MG	n	1.61

Sumário

1 Entrando e saindo com dados do R

- Importação de dados
- Exportando dados

2 Manipulando dados

- Edição
- Criação de subconjuntos
- Criação de filtros
- Ordenação

Subconjuntos

Tabela: Trapalhões

	codinome	nascimento	estado	vivo	altura
1	Didi	1936	CE	s	1.680
2	Dede	1936	RJ	s	1.790
3	Mussum	1941	RJ	n	1.810
4	Zacarias	1934	MG	n	1.610

Subconjuntos

Tabela: Trapalhões

	codinome	nascimento	estado	vivo	altura
1	Didi	1936	CE	s	1.680
2	Dede	1936	RJ	s	1.790
3	Mussum	1941	RJ	n	1.810
4	Zacarias	1934	MG	n	1.610

Sinônimo de indexação.

Subconjuntos: vetores

	1	2	3	4
animal	peixe	cão	gato	hamster

```
> animal<-c( "peixe" ,"cão","gato","hamster")
```

Subconjuntos: vetores

	1	2	3	4
animal	peixe	cão	gato	hamster

```
> animal<-c( "peixe" ,"cão","gato","hamster")  
> animal[2]  
[1] "cão"
```

Subconjuntos: vetores

	1	2	3	4
animal	peixe	cão	gato	hamster

```
> animal<-c( "peixe" ,"cão","gato","hamster")  
> animal[2]  
[1] "cão"  
> animal[c(2,3)]  
[1] "cão" "gato"  
> animal[2:4]  
[1] "cão" "gato" "hamster"
```


Subconjuntos: vetores

```
> animal<-c( "peixe" ,"cão","gato","hamster")
```

Subconjuntos: vetores

```
> animal<-c( "peixe" ,"cão","gato","hamster")  
> animal[c(1,1,2,3)]  
[1] "peixe" "peixe" "cão" "gato"
```

Subconjuntos: vetores

```
> animal<-c( "peixe" ,"cão","gato","hamster")  
> animal[c(1,1,2,3)]  
[1] "peixe" "peixe" "cão" "gato"  
> animal[-2]  
[1] "peixe" "gato" "hamster"
```

Subconjuntos: matrizes e *data.frame*

Matrizes e *data.frames* possuem 2 dimensões.

Subconjunto em uma matriz

`matriz[linha, coluna]`

```
> x<-matrix(1:12,ncol=3)
```

```
> x
```

	[,1]	[,2]	[,3]
[1,]	1	5	9
[2,]	2	6	10
[3,]	3	7	11
[4,]	4	8	12

```
> x[2,2]
```

```
[1] 6
```

Subconjuntos: matrizes e *data.frame*

```
> x[2:4,2:3]
```

	[, 1]	[, 2]
[1,]	6	10
[2,]	7	11
[3,]	8	12

Subconjuntos: matrizes e *data.frame*

```
> x[2:4,2:3]
```

	[, 1]	[, 2]
[1,]	6	10
[2,]	7	11
[3,]	8	12

Todas as colunas de uma linha

```
> x[1,]  
[1] 1 5 9
```

Subconjuntos: matrizes e *data.frame*

```
> x[2:4,2:3]
```

	[, 1]	[, 2]
[1,]	6	10
[2,]	7	11
[3,]	8	12

Todas as colunas de uma linha

```
> x[1,]
```

```
[1] 1 5 9
```

Todas as linhas de uma coluna

```
> x[,1]
```

```
[1] 1 2 3 4
```

Subconjunto: nomes das colunas em *data.frame*

É possível usar o nome da coluna de um `data.frame`.

Seleção do conteúdo de uma coluna

`objeto$coluna`

```
> trapa$COD
[1] Didi Dede Mussum Zacarias
Levels: Dede Didi Mussum Zacarias
> trapa$NASC
[1] 1936 1936 1941 1934
```


Subconjunto: nomes das colunas em *data.frame*

É possível alterar o conteúdo de uma coluna

```
> trapa$VIVO<-c("T", "T", "F", "F")
```

Subconjunto: nomes das colunas em *data.frame*

É possível alterar o conteúdo de uma coluna

```
> trapa$VIVO<-c("T", "T", "F", "F")
```

E criar novas colunas

```
> trapa$idade<- 2014-trapa$NASC
```

	COD	NASC	UF	VIVO	Altura	idade
[1,]	Didi	1936	CE	T	1.68	78
[2,]	Dede	1936	RJ	T	1.79	78
[3,]	Mussum	1941	RJ	F	1.81	73
[4,]	Zacarias	1934	MG	F	1.61	80

Subconjuntos: listas

```
> aluno<-list(nome= "José", idade=20, esportes=c("natação",  
"surfe"))
```

Subconjuntos: listas

```
> aluno<-list(nome= "José", idade=20, esportes=c("natação",  
"surfe"))  
> aluno  
$nome  
[1] "José"  
  
$idade  
[1] 20  
  
$esportes  
[1] "natação" "surfe"
```

Subconjuntos: listas

```
> aluno$esportes  
[1] "natação" "surfe"
```

Subconjuntos: listas

```
> aluno$esportes  
[1] "natação" "surfe"
```

```
> aluno[3]  
$esportes  
[1] "natação" "surfe"
```

Subconjuntos: listas

```
> aluno$esportes  
[1] "natação" "surfe"
```

```
> aluno[3]  
$esportes  
[1] "natação" "surfe"
```

```
> aluno[[3]]  
[1] "natação" "surfe"
```

Subconjuntos: listas

```
> aluno$esportes  
[1] "natação" "surfe"
```


Subconjuntos: listas

```
> aluno$esportes  
[1] "natação" "surfe"  
  
> aluno$esportes[1]  
[1] "natação"
```

Subconjuntos: listas

```
> aluno$esportes  
[1] "natação" "surfe"  
  
> aluno$esportes[1]  
[1] "natação"  
  
> aluno[[3]][1]  
[1] "natação"
```

Sumário

1 Entrando e saindo com dados do R

- Importação de dados
- Exportando dados

2 Manipulando dados

- Edição
- Criação de subconjuntos
- Criação de filtros
- Ordenação

Usando filtros

Um filtro é uma condição lógica que permite selecionar dados.
Para isso é usado os operadores comparação e lógicos.

Operadores lógicos

& &

E

| |

OU

!

Negação

Exemplos

```
> z<-c(1,2,3,4,5,6,7,8)
```

```
> z>3
```

```
[1] FALSE FALSE FALSE TRUE TRUE TRUE TRUE TRUE
```

```
> z[z>3]
```

```
[1] 4 5 6 7 8
```

Exemplos

```
> z<-c(1,2,3,4,5,6,7,8)
```

```
> z>3
```

```
[1] FALSE FALSE FALSE TRUE TRUE TRUE TRUE TRUE
```

```
> z[z>3]
```

```
[1] 4 5 6 7 8
```

```
> z>3 & z<7
```

```
[1] FALSE FALSE FALSE TRUE TRUE TRUE FALSE FALSE
```

```
> z[z>3 & z<7]
```

```
[1] 4 5 6
```

Exemplos

```
> Altura  
[1] 1.85 1.78 1.92 1.63 1.81 1.55  
  
> Sexo  
[1] M M M F F F  
Levels:  F M
```

Exemplos

```
> Altura
[1] 1.85 1.78 1.92 1.63 1.81 1.55
> Sexo
[1] M M M F F F
Levels:  F M

> homens.altos<- Altura >= 1.80 & Sexo== "M"
> homens.altos
[1] TRUE FALSE TRUE FALSE FALSE FALSE
```


Exemplos

```
> homens.altos  
[1] TRUE FALSE TRUE FALSE FALSE FALSE
```

Quantos homens altos existem?

```
> sum(homens.altos)  
[1] 2
```

Proporção de homens altos

```
> mean(homens.altos)  
[1] 0.3333333
```

Exemplos

```
> ilhas
```

	[,1]	[,2]	[,3]	[,4]
[1,]	3	2	0	2
[2,]	1	0	0	2
[3,]	1	1	1	0
[4,]	1	2	2	0
[5,]	1	1	2	2

```
> ilhas>1
```

	[,1]	[,2]	[,3]	[,4]
[1,]	TRUE	TRUE	FALSE	TRUE
[2,]	FALSE	FALSE	FALSE	TRUE
[3,]	FALSE	FALSE	FALSE	FALSE
[4,]	FALSE	TRUE	TRUE	FALSE
[5,]	FALSE	FALSE	TRUE	TRUE

Exemplos

```
> apply(ilhas>1,2,sum)
[1] 1 2 2 3
```

Sumário

1 Entrando e saindo com dados do R

- Importação de dados
- Exportando dados

2 Manipulando dados

- Edição
- Criação de subconjuntos
- Criação de filtros
- **Ordenação**

Ordenação

Ordenação é o ato de se colocar os elementos de uma sequência de informações, ou dados, em uma ordem predefinida.

Funções de ordenação

Ordena o vetor

```
sort(x, decreasing=FALSE)
```

Retorna os índices ordenados

```
order(x, decreasing=FALSE)
```

Exemplos

```
> sort(Altura)
[1] 1.55 1.63 1.78 1.81 1.85 1.92
```

Exemplos

```
> sort(Altura)
[1] 1.55 1.63 1.78 1.81 1.85 1.92

> order(Altura)
[1] 6 4 2 5 1 3
```

Exemplos

```
> sort(Altura)
[1] 1.55 1.63 1.78 1.81 1.85 1.92

> order(Altura)
[1] 6 4 2 5 1 3

> Altura[order(Altura)]
[1] 1.55 1.63 1.78 1.81 1.85 1.92
> Altura.Crescente<-Altura[order(Altura)]
```


Cuidado ao ordenar matrizes/*data.frames*

Tabela: Trapalhões

	codinome	nascimento	estado	vivo	altura
1	Didi	1936	CE	s	1.680
2	Dede	1936	RJ	s	1.790
3	Mussum	1941	RJ	n	1.810
4	Zacarias	1934	MG	n	1.610

Mantendo a relação entre as colunas

```
> order(trapa$COD)
```

```
[1] 2 1 3 4
```

```
> Em.ordem<-order(trapa$COD)
```

Mantendo a relação entre as colunas

```
> order(trapa$COD)
```

```
[1] 2 1 3 4
```

```
> Em.ordem<-order(trapa$COD)
```

```
> trapa.em.ordem<-trapa[Em.ordem,]
```

```
> trapa.em.ordem
```

	COD	NASC	UF	VIVO	Altura
2	Dede	1936	RJ	s	1.79
1	Didi	1936	CE	s	1.68
3	Mussum	1941	RJ	n	1.81
4	Zacarias	1934	MG	n	1.61

Próxima aula

- Ler cap.3 e 4 da apostila.
- Fazer tutorial “Leitura e manipulação de dados” da apostila.
- Fazer exercícios “Logica” e “Manipulando os dados” do Swirl.